Lehrstuhl Angewandte Informatik IV
Datenbanken und Informationssysteme
Prof. Dr.-Ing. Stefan Jablonski

Institut für Angewandte Informatik
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth

Seminar

Vorname Nachname

August 26, 2015
Version: Draft / Final

Universität Bayreuth

Fakultät Mathematik, Physik, Informatik
Institut für Informatik
Lehrstuhl für Angewandte Informatik IV

Titel / Topic

Seminar

Vorname Nachname

1. Reviewer Prof. Dr.-Ing. Stefan Jablonski
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

2. Reviewer Dr. Stefan Schönig
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

Supervisors Stefan Schönig and Lars Ackermann

August 26, 2015

Vorname Nachname
Seminar
Titel / Topic, August 26, 2015
Reviewers: Prof. Dr.-Ing. Stefan Jablonski and Dr. Stefan Schönig
Supervisors: Stefan Schönig and Lars Ackermann

Universität Bayreuth
Lehrstuhl für Angewandte Informatik IV
Institut für Informatik
Fakultät Mathematik, Physik, Informatik
Universitätsstrasse 30
95447 Bayreuth
Germany

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Abstract (different language)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Acknowledgement

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

i really wanna stay at your house

Contents

1 Introduction

1.1 Motivation

Machine learning (ML) has become a pivotal tool in decision-making processes across numerous domains, including healthcare, finance and hiring. However, as these models increasingly influence critical decisions, questions about their fairness and the ethical implications of their use have come to the forefront. Fairness in ML concerns the equitable treatment of individuals or groups, particularly in the presence of sensitive attributes such as gender, age, race, or socioeconomic status. These attributes often correlate with historical inequalities or systemic biases embedded in the data. ML models, designed to optimize accuracy, learn patterns from this data and may inadvertently exploit these unfair patterns to make predictions. While leveraging such patterns can improve predictive performance, it also risks perpetuating or even amplifying existing inequities. A notable example of this occurred when Amazon's AI recruiting tool, which was designed to assist in hiring decisions, exhibited significant gender bias, favoring male candidates due to biases in historical hiring data, leading to its eventual scrapping. [Cha23]

These challenges of fairness and bias are not confined to traditional applications but also extend to the field of predictive business process monitoring (PBPM). Here, ML models play an integral role in making decisions, whether by predicting outcomes, allocating resources, or streamlining operations. Addressing fairness in this context requires careful consideration of bias's dual nature. While biases can lead to discrimination, not all bias is inherently harmful. In some cases, certain biases may be necessary for the model to achieve its intended purpose. For instance, sensitive attributes like gender can carry significant information relevant to specific contexts. In the domain of healthcare, gender differences in biological and hormonal factors are crucial for determining effective treatments and drug prescriptions. However, the same attribute might lead to discriminatory outcomes in unrelated contexts, such as predicting whether a patient's request for treatment will be approved. This duality highlights the complexity of fairness in ML: biases that are helpful in one scenario can become harmful in another.

A significant obstacle in addressing this issue is the inherent opacity of ML models. Many models, especially complex ones such as deep neural networks, operate as

"black boxes" that produce predictions without providing insight into how those predictions are made. This lack of interpretability makes it challenging for human stakeholders and domain experts to identify whether a model's reliance on a sensitive attribute aligns with ethical and operational goals. This presents a dilemma: either leaving potentially harmful biases unchecked to avoid significantly compromising the model's predictive performance, or removing sensitive attributes entirely from the model's input. The latter approach, while intended to prevent discrimination, can lead to suboptimal predictions, especially when the sensitive attributes carry critical information relevant to the task.

1.2 Problem Statement

To address this challenge, this thesis proposes an approach based on knowledge distillation. Knowledge distillation involves transferring the encapsulated knowledge from a complex model to a simpler, more interpretable representation. By applying this technique, the inner workings of a predictive ML model can be visualized and understood by human stakeholders and domain experts, which in turn enables the identification of inherent biases in the model. Specifically for the proposed approach, the distilled representation is modified to remove the unwanted bias by adjusting the way sensitive attributes influence the representation's decision-making process. This modified version of the representation is then used to relabel the training data, creating a refined dataset that reflects the desired fairness characteristics. Finally, the original model is fine-tuned using this new, bias-adjusted training data, allowing it to learn from the corrected representation and ultimately make more equitable predictions.

The goal of this approach is twofold: to make the ML model fairer by eliminating the biases that result in unfair treatment and to maintain its predictive accuracy by preserving helpful ones. By balancing these objectives, the proposed methodology seeks to create models that are not only effective but also aligned with given ethical standards and societal expectations.

1.3 Thesis Outline

Building on the problem statement, the second chapter provides insight into existing research concerning the field of bias reduction and fairness in PBPM, while highlighting the gaps this thesis aims to address.

The third chapter provides background information essential for understanding the research. It introduces PBPM, explains the ML models used in this thesis and discusses fairness in ML.

The fourth chapter details the methodology employed in the thesis. It describes the process of data gathering and preprocessing, with a focus on handling bias causing attributes. The chapter then explains how the initial ML model is trained, followed by the application of knowledge distillation to create an interpretable representation of the model, which can then be modified to address unwanted biases. The methodology further includes how the predictions of the modified representation can be used to finetune the model to improve fairness, while maintaining accuracy.

TODO The fifth chapter presents the evaluation of the proposed approach. This chapter defines the metrics used for evaluation and includes empirical results from multiple case studies.

TODO The final chapter concludes the thesis by summarizing the findings and discussing their implications for fairness in ML. It acknowledges the limitations of the research and offers suggestions for future work.

2 Related Work

In recent years, the field of fairness in machine learning has gained significant attention, aiming to address biases and ensure equitable outcomes in predictive systems. Numerous approaches to quantifying and achieving fairness have been proposed, targeting various steps in the ML pipeline. These methods are too plentiful to go over in this thesis, but an overview can be found in [CH24].

Comparatively, the existing literature related to fairness specifically in PBPM is quite limited, although many foundational concepts and challenges concerning fairness can be transferred from general ML, as done in [DAFST22]. Accordingly, the next section will focus on reviewing approaches that have explicitly been adapted into the field of PBPM.

2.1 Fairness in Predictive Business Process Monitoring

[QA19] represents one of the initial efforts to incorporate fairness into PBPM. This approach employs discrimination-aware decision trees [KCP10], which extend traditional decision trees by adding fairness constraints during the split-selection process. These constraints penalize splits influenced by sensitive attributes, ensuring that decisions are less biased. To further enhance fairness, a relabeling technique is employed, where outcome classes of the leaf nodes are adjusted to reduce the discriminatory impact of sensitive attributes with minimal loss in predictive accuracy. This technique is similar to our methodology as described in section **??**, there is however a key difference: The relabeling in this method is performed automatically, based on the assumption that all bias is undesirable and should be eliminated. In contrast, our approach emphasizes human stakeholder involvement to decide which biases are unwanted, allowing for the preservation of nuanced, context-dependent correlations.

In [LP24], biases in predictive models are being adressed by leveraging adversarial learning [Goo+14], a method where two models are trained simultaneously: a predictor and an adversary. The predictor model focuses on accurate outcomes, while an adversary model tries to identify sensitive attributes based on the output of

the predictor. The predictor is penalized when the adversary succeeds, encouraging fairness by reducing reliance on these attributes. While effective at reducing bias, this penalty-based approach lacks context sensitivity, treating all biases as inherently undesirable. This indiscriminate removal of biases may unintentionally eliminate meaningful correlations, thereby diminishing the predictive model's utility in specific scenarios where such relationships are crucial.

Lastly, [PDV24] focuses on ensuring independence between predictions and sensitive group memberships using a composite loss function for training the ML model. Compared to traditional loss functions, which seek to only optimize the predictive performance of a ML model, this composite loss function incorporates integral probability metrics [PZ19], in order to find a balance between accuracy and fairness. However, similar to the other approaches presented before, the uniform elimination of biases may inadvertently remove desirable correlations, potentially affecting the utility of the predictive model in certain contexts.

2.2 Conclusion

Current approaches to fairness in PBPM have made significant strides but remain somewhat limited by their automated and uniform treatment of bias. A shift toward an approach that puts more control into the hands of stakeholders presents a promising direction for achieving more context-sensitive models, that achieve better performance while still making context-based fair decisions.

3 Background

This chapter will go over definitions and background knowledge necessary for understanding the methodology and evaluation presented in later chapters. It will adress fundamentals of PBPM, go over the basics of neural networks and decision trees, and explain how fairness is quantified in this thesis.

3.1 Predictive Business Process Monitoring

Predictive Business Process Monitoring (PBPM) is a field that focuses on analyzing and forecasting the progression of ongoing business processes based on historical data. **Business processes** represent structured sets of activities or tasks undertaken by organizations to achieve specific objectives, such as processing customer orders, managing inventory or onboarding new employees. A specific execution or instance of the overall business process is refered to as a **case**. For example, in an order fulfillment process, each order corresponds to a separate case.

3.1.1 Business Process Model and Notation

In order to provide visualization for stakeholders, business processes are often portrayed in the form of **process models**. These capture the sequence and flow in a formalized representation and are generally created using the standardized **Business Process Model and Notation**, which we will also use a simplified subset of [Whi04]:

- **Activities** are represented as rectangles with rounded corners, denoting the individual tasks or operations within the process.

- **Gateways** are depicted as diamonds, used to control divergence and convergence in the workflow. These indicate either decision points or the merging of paths. In this thesis, decisions are considered to be exclusive, so only one of the paths can be chosen.

- **Events** are shown as circles, representing the initiation and completion of the process. Specifically, the start event, depicted as a circle with a narrow border,

marks the beginning of the process and the end event, depicted as a circle with a bold border, signifies the process's conclusion.

- **Sequence Flow** is represented by lines with arrowheads and is used to show the order that activities will be performed in.

3.1.2 Event Log Data

During the execution of business processes, data is recorded in the form of event logs. In order to formally define the structure of this data, let A be the universe of all process activities, C the universe of case IDs, T the universe of possible timestamps and $S_1, ..., S_m$ the universes of the m attributes associated with the process. In this thesis, we will assume all relevant attributes to be case-level. Case-level attributes are static, which means that their values won't change during the execution of a process instance. An example for such an attribute could be the nationality or gender of a patient in a hospital.

In an event log, executed activities as stored as events. Such an **event** e can be described as a tuple $e = (a, c, s, t)$. In this tuple, $a \in A$ is the activity that has been executed. For ease of notation, we will use the shortcut $\mathit{activity}(e) = a$ to access the activity a belonging to event e. $c \in C$ is the case ID, linking the event to a specific process instance. s is a tuple $s = (s_1, ..., s_m)$ with $s_i \in S_i, \forall i \in \{1, ..., m\}$, containing the values of each attribute. $t \in T$ is the timestamp denoting when the event occurred.

The sequence of all events $e_1, ..., e_l$ belonging to a single case c, ordered by their timestamps, is called the **trace** of c with length l. Considering that a trace describes the life-cycle of a single case c, all events belonging to a trace have the same attribute values s. A **prefix** with length l' is a portion of such a trace with length l, consisting of the first l' events, with $l' < l$. Lastly, the **event log** is a finite collection of traces belonging to the same process.

3.1.3 Next Activity Prediction

When utilizing these event logs for PBPM, there are multiple tasks one could consider taking on, such as predicting the remaining time of a case or determining its final outcome. These predictions play a crucial role in enabling organizations to optimize operations, allocate resources effectively, and preempt potential issues. In this thesis, we will take a closer look specifically at **next activity prediction**, where the objective is to forecast the subsequent activity in an ongoing case. Formally, given a prefix

$e_1, ..., e_{l'}$ of observed events, the goal is to predict *activity*$(e_{l'+1})$, the activity label of the next event $e_{l'+1}$.

3.2 Black Box vs. White Box

Within the field of PBPM, numerous traditional machine learning and deep learning architectures have been employed to develop effective predictors for next activity prediction [RMVL21]. A fundamental distinction among these approaches lies in their categorization as either black box or white box models.

Black box models, such as neural networks, excel in capturing complex patterns and relationships within data but provide limited transparency into their decision-making processes. In contrast, **white box** models, like decision trees or linear regression, prioritize interpretability and simplicity, enabling stakeholders to understand and trust the reasoning behind predictions. However, this may come at the cost of reduced predictive performance for complex tasks. [Rud19]

The difference between black and white box models will become even clearer in the following sections 3.3 and 3.4, as we will look at feedforward neural networks as an example for a black box model and decision trees as an example for a white box model.

3.3 Neural Networks

In this thesis, **feedforward neural networks** (FNN) will be used as the main predictor for the task of next activity prediction. While these are not inherently designed for sequence data, such as traces in an event log, their simplicity and ease of training still makes them an attractive choice. Additionallly, the focus of this work doesn't lie on achieving maximum predictive performance, but rather the conceptual demonstration of our approach. Moreover, FNNs have been employed in related work [LP24] within PBPM, demonstrating their viability for similar tasks. Nevertheless, since our methodology described in chapter 4 is not specific to using FNNs, it is entirely feasible to adapt our approach to other deep learning architectures.

3.3.1 Perceptron

In order to understand FNNs, it is helpful to begin with the basic computational unit in a neural network, the perceptron. The perceptron processes an input vector

$x = (x_1, ..., x_n)$ in a way that is designed after biological neurons. It combines the input x linearly with a weight vector $w = (w_1, ..., w_n)$ and adds a bias b. The resulting sum is then passed to an activation function σ, which determines the final output of the perceptron. This activation function typically introduces non-linearity into the computation, such as the ReLU or sigmoid function. [**perceptron**]

All in all, the perceptron calculates its scalar output y for the input x as:

$$y = \sigma\left(\left[\sum_{i=1}^{n} w_i \cdot x_i\right] + b\right) \tag{3.1}$$

3.3.2 Network Architecture

The scalar output of a perceptron can only be used for basic binary classification or regression problems. Even then, due to the simple construction of the perceptron, its effectiveness is highly limited, depending on the complexity of the task. [**perceptron_limited**] FNNs extend the perceptron model by organizing multiple perceptrons into a structured, layered network, enabling them to handle complex tasks.

An FNN consists of three distinct types of layers, ordered sequentially:

- Input layer: The input layer acts as the interface between raw input data and the network. Each perceptron in this layer corresponds to one feature of the input vector $x = (x_1, ..., x_n)$. The input layer does not perform any transformations, it merely transfers the raw input values to the next layer.

- Hidden layers: Hidden layers are where the network performs the majority of its computations. Each hidden layer contains a set of perceptrons that process data received from the previous layer. These layers are responsible for learning complex representations of the input data, enabling the network to capture intricate patterns and relationships.

- Output layer: The output layer produces the final predictions of the network. For multiclass classification tasks, such as next activity prediction, the layer includes one perceptron for each class.

The number of hidden layers, the number of perceptrons per layer, and the choice of activation functions collectively define the architecture of the network. These design choices are critical in determining the model's ability to handle the task at hand while balancing computational complexity and capacity.

3.3.3 Feedforward Algorithm

With the network architecture defined, the next step is to understand how input data is processed as it passes through the various layers. This process is governed by the **feedforward algorithm**.

The algorithm begins at the input layer, where the input $x = (x_1, ..., x_n)$ is passed directly to the next layer without any transformations. From there, each perceptron in the subsequent hidden and output layers performs the same operation as provided in formula 3.1, using the outputs from the previous layer as inputs. Formally, the activiation $y_j^{(l)}$ of the j-th perceptron in the l-th layer is computed from the outputs $y_1^{(l-1)}, ..., y_m^{(l-1)}$ of the previous layer, using weight vector $w^{(l)} = (w_{1,j}^{(l)}, ..., w_{m,j}^{(l)})$, bias $b_j^{(l)}$ and activation function $\sigma_j^{(l)}$:

$$y_j^{(l)} = \sigma\left(\left[\sum_{i=1}^{m} w_{i,j}^{(l)} \cdot y_i^{(l-1)}\right] + b_j^{(l)}\right) \tag{3.2}$$

Once the algorithm completes, the values produced by the perceptrons in the output layer are passed through a softmax activation function, which transforms them into a probability distribution across all possible classes. The predicted class corresponds to the one with the highest probability.

This strictly unidirectional flow of data ensures that no feedback loops are introduced, maintaining computational simplicity. Each layer's output is used exclusively as the input for the next layer, making the process of computation straightforward and efficient. However, the interplay of numerous weights and biases across multiple layers, combined with the non-linear transformations introduced by activation functions, makes it difficult to intuitively understand or trace how specific inputs lead to specific outputs. This lack of transparency is why we consider FNNs to be black box models.

3.3.4 Backpropagation Algorithm

3.4 Decision Trees

Similarly to FNNs, we will also use **Decision Trees** (DT) as a predictor for next activity prediction. One of their most significant strengths lies in their transparency and interpretability, especially when compared to more opaque models like FNNs. And

despite their lack of an inherent mechanism for handling sequential dependencies, decision trees have been effectively employed in related work [QA19] within the PBPM domain, making them a suitable choice for in this thesis.

3.4.1 Tree Structure and Key Components

In order to understand DTs, we will first go over the structure and key components of a tree in general. A tree T is a special type of directed graph, defined as $T = (V, E)$, where V is the finite set of **nodes** and $E \subset V \times V$ is the set of directed **edges**. When two nodes $u, v \in V$ are connected by an edged $(u, v) \in E$, we will call u the **parent** of v and likewise v the **child** of u. Similarly, when there exists a sequence of edges $(u, w_1), (w_1, w_2), ..., (w_{n-1}, w_n), (w_n, v) \subset E$, u is an **ancestor** of v and v is a **descendant** of u.

Within the tree, nodes are classified into either **internal nodes**, when they have at least one child, or **leaf nodes**, when they have no children. Additionallly, a node r without parents, formally satisfying $\forall v \in V : (v, r) \notin E$, denotes the **root** of the tree. Accordingly, the **subtree** rooted at a node v refers to the tree $T_v = (V_v, E_v)$, where $V_v \subset V$ is the subset containing all descendants of v and $E_v \subset E$ the edges between V_v.

Lastly, unlike a general directed graph, a tree must satisfy several strict structural properties: [**trees**]

- **Single Root:** There has to be a unique root node $r \in V$.
- **Connectedness:** Every node $v \in V$ must be a descendant of root node r.
- **Single Parent:** Every node $v \in V$ except the root node r must have exactly one parent $u \in V$.

Furthermore, we will work exclusively with binary trees. These have the additional restriction, that every node must have either exactly two or none children. When a node has two children, we will assume a fixed distinction into a left child an a right child.

3.4.2 Decision Process

Having presented the necessary definitions, we will now take a closer look at how DTs work internally. In this thesis, input data for DTs will have the same shape as

the input for FNNs. Therefore a sample $x = (x_1, ..., x_n)$ is a vector containing n numerical features. When predicting the next activity y for such a sample x, the nodes of the DT serve different roles:

The root node is the starting point for all decision processes. Internal nodes posess a feature index i and a threshold value t. These internal nodes represent decision points, which examine whether the i-th feature x_i of the sample x is greater than the threshold t. Depending on the answer, the sample x is then passed onto either the left or right child. This process continues recursively until the sample reaches a leaf node. Leaf nodes contain a label y, representing the outcomes of the decision process, in our case determining the predicted next activity.

Since the structure of a DT explicitly represents the decision-making process, it becomes immediately clear why they are widely regarded as white box models. Each path from the root to a leaf corresponds to a set of human-readable decision rules, enabling stakeholders to identify important features and assess the rationale behind each decision.

3.4.3 Training Process

Now that we understand how a DT makes predictions, we turn our focus to the training process—how a DT is built from data. Let S be the collection containing the training data and $|S|$ the amount of elements contained in S. We assume the elements of S to be tuples of the form (x, y), where $x = (x_1, ..., x_n)$ is a sample input and y the corresponding target output label. The training process aims to construct a structure that effectively partitions S in a way, such that the target labels y within the partitions are as homogeneous as possible. This is achieved by selecting the feature and threshold for each split that maximize the "purity" of the resulting subsets.

A common metric used to evaluate this purity in data sets is **Gini impurity**. The Gini impurity measures the likelihood of incorrectly classifying a randomly chosen element, if it were labeled according to the distribution of labels. Intuitively, it quantifies how "mixed" the labels are within S. Mathematically, for k different target labels, where p_i is the proportion of samples in S having the i-th target label, the Gini impurity is defined as:

$$G(S) = 1 - \sum_{i=0}^{k} p_i^2 \tag{3.3}$$

The Gini impurity $G(S)$ ranges from 0 to $1 - \frac{1}{k}$. A value of $G(S) = 0$ indicates that S is perfectly pure, with all samples having a single target label, while $G(S) = 1 - \frac{1}{k}$

represents a maximally impure dataset where samples are evenly distributed among all k target labels. Generally, a lower $G(S)$ implies a higher purity. [**gini**]

The training process begins at the root node, which is associated with the entire training dataset S. At each step, the algorithm evaluates all possible splits of the data by testing every feature x_i and various thresholds. Thresholds for splitting are derived directly from the values in the dataset associated with the node, ensuring that every threshold capable of separating samples into distinct subsets is tested. Specifically, given a feature x_i and a sorted list of its values in SS, potential thresholds are chosen as the midpoints between consecutive unique values. The split that minimizes the impurity of the resulting subsets is chosen. Formally, if splitting S into S_1 and S_2 results in Gini impurities $G(S_1)$ and $G(S_2)$, the split is selected to minimize the weighted Gini impurity G_{split}:

$$G_{split}(S_1, S_2) = \frac{|S_1|}{|S|} G(S_1) + \frac{|S_2|}{|S|} G(S_2) \tag{3.4}$$

Once the best split is determined, two child nodes are created, and each is associated with one of the datasets S_1 and S_2 resulting from the split. This process is recursively repeated for each child node until a stopping criterion is met, such as reaching a maximum tree depth, achieving a minimum number of samples per leaf, or reducing impurity to a negligible level.

3.4.4 Cost Complexity Pruning

Depending on the selected stopping criterion, the final DT may include subtrees with numerous nodes that contribute little to its predictive value. To address this, pruning is a common technique used to simplify decision trees by removing unnecessary branches, ultimately improving the tree's performance and generalizability. Additionallly, the reduced amount of nodes will make the DT more easily interpretable for human stakeholders.

One effective approach to pruning is **cost-complexity pruning**, which balances the trade-off between a tree's complexity and its predictive accuracy. [**ccp**]

3.5 Fairness

As machine learning systems are increasingly deployed in high-stakes applications, ensuring fairness has become a critical concern. Both neural networks and decision trees, while powerful tools for prediction, must be scrutinized for potential biases in their outcomes. This concern underscores the importance of having quantitative

measures for assessing and comparing models to ensure that they meet fairness criteria alongside performance metrics.

Quantitative group fairness metrics provide a systematic way to evaluate whether a machine learning model behaves equitably across different groups within the population. In this thesis, we focus on one of the most prevalent fairness metrics, **demographic parity**. [**dem_parity**]

4 Methodology

This chapter will outline the complete pipeline of our methodology. It describes the process of data gathering and preprocessing, with a focus on handling bias causing attributes. The chapter then explains how the initial ML model is trained, followed by the application of knowledge distillation to create an interpretable representation of the model, which can then be modified to address unwanted biases. The methodology further includes how the predictions of the modified representation can be used to finetune the original model in order to improve fairness, while maintaining accuracy.

4.1 Data Gathering

Public real life event logs containing sensitive attributes are hard to come by. Additionally, in order to properly highlight the capacity our method, we require the data to have sensitive attributes that cause both positive and negative bias respectively. Although [**simulated_logs**] recently published simulated event logs, seeking to address the scarcity of fairness-aware datasets in PBPM, those event logs didn't show the structure we were looking for in this thesis. Therefore we will create the necessary data ourselves, either by simulating a process model or by enriching a real life event log with sensitive attributes.

4.1.1 Data Simulation

4.1.2 Data Enrichment

4.2 Data Processing

4.3 Training the Model

4.4 Knowledge Distillation

4.5 Modification of the Decision Tree

4.5.1 Cutting Branches

4.5.2 Retraining Subtrees

4.6 Finetuning of the Model

Deep learning generally outperforms traditional machine learning [Kra+21]

5 Evaluation

5.1 Cancer Screening

5.2 Hospital Billing

5.3 BPI Challenge 2012

5.4 Ablation

6 Conclusion

Bibliography

[CH24] Simon Caton and Christian Haas. „Fairness in machine learning: A survey“. In: *ACM Computing Surveys* 56.7 (2024), pp. 1–38 (cit. on p. 5).

[Cha23] Xinyu Chang. „Gender Bias in Hiring: An Analysis of the Impact of Amazon’s Recruiting Algorithm“. In: *Advances in Economics, Management and Political Sciences* 23 (2023), pp. 134–140 (cit. on p. 1).

[DAFST22] Maria De-Arteaga, Stefan Feuerriegel, and Maytal Saar-Tsechansky. „Algorithmic fairness in business analytics: Directions for research and practice“. In: *Production and Operations Management* 31.10 (2022), pp. 3749–3770 (cit. on p. 5).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. „Generative adversarial nets“. In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 5).

[KCP10] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. „Discrimination Aware Decision Tree Learning“. In: *2010 IEEE International Conference on Data Mining*. 2010, pp. 869–874 (cit. on p. 5).

[Kra+21] Wolfgang Kratsch, Jonas Manderscheid, Maximilian Röglinger, and Johannes Seyfried. „Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction“. In: *Business & Information Systems Engineering* 63 (2021), pp. 261–276 (cit. on p. 18).

[LP24] Massimiliano de Leoni and Alessandro Padella. „Achieving Fairness in Predictive Process Analytics via Adversarial Learning (Extended Version)“. In: *arXiv preprint arXiv:2410.02618* (2024) (cit. on pp. 5, 9).

[PDV24] Jari Peeperkorn and Simon De Vos. „Achieving Group Fairness through Independence in Predictive Process Monitoring“. In: *arXiv preprint arXiv:2412.04914* (2024) (cit. on p. 6).

[PZ19] Victor M Panaretos and Yoav Zemel. „Statistical aspects of Wasserstein distances“. In: *Annual review of statistics and its application* 6.1 (2019), pp. 405–431 (cit. on p. 6).

[QA19] Mahnaz Sadat Qafari and Wil Van der Aalst. „Fairness-aware process mining“. In: *On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21–25, 2019, Proceedings*. Springer. 2019, pp. 182–192 (cit. on pp. 5, 12).

[RMVL21] Efrén Rama-Maneiro, Juan C Vidal, and Manuel Lama. „Deep learning for predictive business process monitoring: Review and benchmark“. In: *IEEE Transactions on Services Computing* 16.1 (2021), pp. 739–756 (cit. on p. 9).

[Rud19] Cynthia Rudin. „Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead“. In: *Nature machine intelligence* 1.5 (2019), pp. 206–215 (cit. on p. 9).

[Whi04] Stephen A White. „Introduction to BPMN“. In: *Ibm Cooperation* 2.0 (2004), p. 0 (cit. on p. 7).

List of Figures

List of Tables

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

Bayreuth, August 26, 2015

Vorname Nachname